DOI:
10.1111/coin.12129

Document Version
Peer reviewed version

Computational Intelligence, Volume 000, Number 000, 0000

Towards Personalised and Adaptive QoS Assessments via Context Awareness

LINA BARAKAT

King's College London, UK

lina.barakat@kcl.ac.uk

PHILLIP TAYLOR, NATHAN GRIFFITHS

University of Warwick, UK

{*Phillip.Taylor, Nathan.Griffiths*}*@warwick.ac.uk*

ADEL TAWEEL, MICHAEL LUCK, SIMON MILES

King's College London, UK

{*adel.taweel, michael.luck, simon.miles*}*@kcl.ac.uk*

QoS (quality of service) properties play an important role in distinguishing between functionally-equivalent services and accommodating the different expectations of users. However, the subjective nature of some properties and the dynamic and unreliable nature of service environments may result in cases where the quality values advertised by the service provider are either missing or untrustworthy. To tackle this, a number of QoS estimation approaches have been proposed, utilising the observation history available on a service to predict its performance. Although the context underlying such previous observations (and corresponding to both user and service related factors) could provide an important source of information for the QoS estimation process, it has only been utilised to a limited extent by existing approaches. In response, we propose a context-aware quality learning model, realised via a learning-enabled service agent, exploiting the contextual characteristics of the domain in order to provide more personalised, accurate and relevant quality estimations for the situation at hand. The experiments conducted demonstrate the effectiveness of the proposed approach, showing promising results (in terms of prediction accuracy) in different types of changing service environments.

Key words: context awareness, service behaviour change, personalisation, quality value learning.

1. INTRODUCTION

Service-oriented computing (SOC) is a suitable paradigm for low-cost, time-effective application development in heterogenous environments, permitting a flexible approach to building complex distributed applications via the integration of loosely-coupled computational entities, offered as *services* by external providers. Web services are a commonly adopted implementation of SOC services, which can be advertised, located, and composed over the web using standards like WSDL, UDDI and BPEL, respectively.

Services advertised by different providers can overlap in their functional capabilities, but offer varying quality of service (QoS) levels. Such QoS properties, thus, play an essential role in differentiating between functionally equivalent services and accommodating different user needs. However, in open and heterogenous service settings, the QoS information available by service providers might be missing or untrustworthy for several reasons, either intentional or unintentional. Specifically, service providers, being autonomous and self-interested, may choose not to fulfil their quality promises (e.g. announce false capabilities to attract more customers). Even with cooperative providers, the quality estimates of services could still frequently change due to other factors (e.g. network and hardware problems, changes in the service implementation, etc). Moreover, in many domains, it might be difficult (or simply not possible) for the provider to guarantee specific values for a quality property, because of its subjective nature or dependency on user-related factors.

Consequently, a number of efforts focus on learning more accurate estimation

of service quality values, based on the data available regarding the service's past performance (Aschoff and Zisman, 2011; Amin et al., 2012; Barakat et al., 2014). Trust and reputation learning mechanisms have also been considered for the purpose of assessing service quality attributes (Maximilien and Singh, 2005; Xu et al., 2007; Malik and Bouguettaya, 2009). Many models exist in which reputation is derived from direct experience of clients with the service and third party recommendations, with numerical or probabilistic representations for reputation (Teacy et al., 2005, 2012; Sabater-Mir and Sierra, 2001; Huynh et al., 2006). Such learning approaches, however, rely on data recency to account for potential changes in the service's behaviour (with respect to the QoS properties). That is, newer service observations are favoured, while older ones are eventually forgotten, without accounting for the circumstances under which the observations were collected. As a result, these approaches neglect important evidence for detecting the occurrence of a change (essential to ensure that only relevant data is captured in the learning process), and ignore situations where old observations may become relevant again (i.e. when a previously encountered service behaviour reappears). Moreover, the observations upon which the quality learning is based are usually assumed to be objective, and thus the predictions produced do not account for a user's particular situation.

In response, we propose enriching service observations with contextual information, and exploiting such information during QoS learning [1]. By context, we refer to any conditions and circumstances that may affect the perception of a quality value by a service user, which could be either related to the user itself (user context) (Shao

[1] A preliminary version of this work appears as a short paper in the 2015 International Conference on Service-Oriented Computing.

et al., 2007; Zheng et al., 2011), or related to the service (service context) (Lin et al., 2012). In particular, our main contributions can be summarised as follows:

- Exploiting user contextual clues to provide personalised quality value assessments that are more tailored towards the client's situation/needs;
- Accounting for service-related circumstances under which past service interactions took place to better assess which of these past interactions are relevant to consider for the quality assessment at hand. In particular, we exploit service contextual clues to judge to what degrees the situations of past service provisions match a new situation, thus achieving better adaptivity in the face of changing service behaviour;
- Allowing older interactions, collected under circumstances comparable to current circumstances, to be predictors of the current service behaviour. This provides supplementary information to recent interactions, especially valuable in the case where the latter are not yet sufficient to make accurate quality predictions.

The rest of the paper is organised as follows. A motivating scenario is presented in Section 2, followed by an elaboration on how context can be recorded in Section 3. Our formal context model is presented in Section 4. Sections 5 and 6 present the context-aware QoS learning model and the experimental results, respectively. Finally, related work and conclusion are discussed in Sections 7 and 8, respectively.

2. MOTIVATING SCENARIO

In this section, we present a number of examples illustrating the importance and benefits of context awareness when assessing the quality characteristics of services.

The examples in this motivating scenario are based on the electronic marketplace for food services (Barakat et al., 2014), developed in the context of the DIET4Elders [2] project, a three year programme including user evaluation of a developed platform. Aside from the marketplace, most details of that platform are not relevant to the research output described here, so not included in the paper. The marketplace provides a rich pool of information enabling the selection of reliable and nutritionally enriched food services for users (older adults in particular), meeting their dietary constraints and different food preferences.

2.1. Importance for Capturing Relevant Evidence

User Context. Consider a scenario in which a user wants to order a meal for dinner. For this purpose, she contacts a food-specialised broker to search for a suitable meal. The broker has access to information about a pool of meal delivery services that are offered by various food providers (which are registered with the broker). The information includes the meal options available, along with their various characteristics, which are either indicated directly by the provider (e.g. the *price* and *ingredients* of a meal), or derived by the broker from previously collected observations on the meals (e.g. utilising past user ratings to assess a meal's *taste*, *smell*, *texture*, *presentation*, *delivery time*, etc.) for the reasons stated in Section 1.

Assume that the user has no preference towards any particular ingredient, but suffers from chewing and swallowing problems, and therefore requires the meal to be of tender texture. She is also interested in a fast meal delivery (less than 40 minutes from the time of order).

[2] http://www.diet4elders.eu/en

Given a candidate meal option (denoted as service s_1), the broker thus needs to assess its corresponding texture and delivery time in order to determine its suitability for the user. Suppose for simplicity that the previous observations on service s_1 are as depicted in Table 1 and Table 2, for texture and delivery time, respectively, and that these observations are collected within the same timespan (i.e. $t_1 \simeq t_2 \simeq t_3$).

Since the perception of food texture could be affected by the presence of chewing and swallowing difficulties, the rating of user 1, which shares similar dysphagia conditions with the current user, should have the highest impact on texture assessment at hand, despite the availability of two other ratings. Similarly, for delivery time (normally affected by the user's location), the reviews indicated by user 1 and user 3 should be regarded as the most relevant for the current user (who also lives in an urban location), while the contribution of that of user 2 should be minimal. That is, service $s1$ should be considered satisfactory for the current user in terms of delivery time despite the rating of user 2 (which is irrelevant for the current situation). Note that ignoring the user's context and simply calculating the average of all past observations would yield an estimated delivery time of 60 minutes, unsatisfactory for the user.

[Table 1 about here.]

[Table 2 about here.]

Service Context. Now consider a similar food ordering scenario where a user is interested in a meal that is highly rated in terms of taste (e.g. at least a 4-star rating). Again, the broker here needs to assess the taste property of each candidate meal option. Assume one such option is service s_2, with the past user ratings available

for this service being depicted in Table 3. Given the mostly good rating history for service s_2, it may be considered suitable for the current user. However, when analysing its provision context, it could be noted that the service exhibited a change in recipe after time step t_{20}, occurring, for instance, due to a change in the head chef, or the temporary unavailability of some ingredients (e.g. some ingredients might not be available at winter time). Such a change could affect many aspects of the meal, including taste, making previous user observations under the old recipe less (or no longer) relevant under the new one. Hence, service s_2 should not be considered satisfactory for the current user (due to being assigned low ratings after the recipe change).

[Table 3 about here.]

2.2. Importance for Tackling Limited Evidence

Assume that, based on another user's preferences, the broker needs to assess the taste of service s_2 at time step t_{61}, and that this service has switched again to *recipe 1* at time step t_{60}. Given this, the *recency* of experiences of this service under *recipe 1* is not sufficient to guide the broker's assessment. However, the broker could exploit the observation history available for service s_2 prior to time step t_{21} under similar recipe (see Table 3) for this purpose. That is, window $[t_1, t_{20}]$ of the ratings on taste available for service s_2 is a useful source of information for the current assessment of taste for this service.

[Figure 1 about here.]

3. CONTEXT DERIVATION FROM PROVENANCE DATA

User-related circumstances can be collected either directly from users and their respective requests, or via utilising an appropriate monitoring platform. For example, in the context of DIET4Elders project, an ambient assisted living infrastructure is built and deployed to collect various data about the user's context (Taweel et al., 2014, 2016).

Providers are the obvious and an important source of service-related circumstances data considered in this paper, as it is a record of how they provided a service. The PROV standard (W3C, 2013), published by W3C as a standard for interoperable provenance, provides a suitable solution for recording information on various contextual circumstances underlying a service provision. A PROV document describes in a queryable form the causes and effects within a particular past process of a system (such as agents interacting, the execution of a program, or enactment of a physical world process), as a directed graph with annotations. A visualisation of such a graph, showing PROV's key elements, is shown in Figure 1. In summary, an *activity* is something that has taken place, making *use* of or *generating entities*, which could be data, physical or other things. *Agents* are parties that were responsible for (*associated with*) activities taking place, and one agent may have been *acting on behalf of* another in this responsibility. Activities, entities and agents (graph nodes) may be annotated with key-value *attributes* describing features that the elements had. The contents of a provenance graph can be collated from data recorded by a set of independent agents, and data can be queried using a standard mean, e.g. by

SPARQL[3] queries. An example provenance graph illustrating the user and service related contextual circumstances presented in our motivating scenario is depicted in Figure 2.

[Figure 2 about here.]

Methods and guidelines for incorporating provenance capture into applications have been developed (Miles et al., 2011; Curcin et al., 2014). These set out how to determine what items of data and records of activities, and causal relationships between them, should be captured at what point and through applying which patterns to adapt the application design and implementation. This is not a trivial topic, as it exposes issues such as how to appropriately capture sensitive data in provenance, how to expose what occurs in legacy components that cannot be adapted, how to capture provenance in anticipation of future questions not yet identified, and so on. However, the grounding principles have been solidly investigated and articulated by this work.

Moreover, incentivisation mechanisms have also been investigated to encourage providers to undergo such changes in their processes and release the provenance data needed. Further elaboration on such incentives can be found in Barakat et al. (2016).

4. CONTEXT MODEL

The quality characteristics offered by a service at a particular moment are not necessarily fixed (as usually assumed by existing quality models of services), but may be dependent on the user situation (user context) under which the service pro-

[3] http://www.w3.org/TR/sparql11-overview/

vision happens. For example, as illustrated in the scenario above, such situational information may include the user's location, the user's medical condition, etc. Furthermore, over time, a service may experience a drift in its behaviour caused by changes in service-related circumstances (service context), which could be either periodic (e.g. a change in a food service's recipe with season) or non-periodic (e.g. a rare event such as a sudden server crash). These service-side changes may lead to a corresponding drift in the perception of service characteristics by users. Figure 3 illustrates such dependency of quality characteristics on contextual features, considering for simplicity: one quality attribute q of a service with two possible outcomes v_q^1 and v_q^2; two user-related contextual attributes c_u^1 and c_u^2; and one service-related contextual attribute c_s, with value v_s^1 at time step t_1 and value v_s^2 at time step t_2.

[Figure 3 about here.]

4.1. Ontological Dimension

Defining and representing contextual knowledge (both user context and service context) is fundamental for facilitating the reasoning about and utilisation of this knowledge. This can be achieved via the use of ontologies, which have proven to be effective in establishing standard models, taxonomies, vocabularies and domain terminology (Stephan et al., 2007). Ontologies describing relevant contextual information are domain dependent, and can be defined by domain experts. For example, a semantic knowledge base is developed in the context of the DIET4Elders project, capturing extensive dietary information of both users and food services (Taweel et al., 2014; Barakat et al., 2014; Taweel et al., 2016). With regards to users, this information includes personal information, clinical information (details of health

problems that are likely to affect food intake and perception, e.g. poor dentition, problems with digestion, etc.), and dietary information (details of texture modification prescriptions, allergies, intolerances, religious and cultural exclusions, etc.). With regards to food providers and their services, the information captures various aspects including full description of offered meal variants, recipes, respective nutrients, types of delivery, geographical coverage, etc. This semantic knowledge base provides a rich source of contextual features (of both users and providers) that are relevant, for instance, in the domain of our motivating scenario.

We make no assumptions in our model about any specific ontological knowledge used, and leave this to the application domain. Instead, we focus on the formal representation of the contextual features, so that our model is kept generic and applicable to a wide range of domains.

[Table 4 about here.]

4.2. Formal Model

Knowledge of context information that is relevant for the provision of a service can be formally modelled as a tuple $(Q, C^u, C^s, ctx_u, ctx_s, dom)$, detailed below (see Table 4 for the notation used throughout the paper).

Q is the set of the quality of service attributes of the service. These attributes can be generic, such as price and response time, or domain-dependent, representing specific features and metrics of a particular domain, such as the taste and presentation of a food service. For example, $Q = \{\text{price}, \text{response time}, \text{taste}, \text{presentation}, ...\}$.

C^u and C^s are the sets of attributes characterising a user's context and the service's context, respectively, and which are expected to affect the quality values

delivered by the service. These attributes are shared among similar types of services, and can be identified by domain experts. Note that our approach does not assume perfect knowledge of contextual attributes, but rather utilises what is available for the purpose of improving the QoS prediction process.

$ctx_u : Q \rightarrow 2^{C^u}$ is a quality attribute's user context function, which maps each quality attribute $q \in Q$ to the set of user-related context attributes that may affect its values, such that $c_u \in ctx_u(q)$ indicates that context attribute c_u could have an impact on the perception of quality attribute q by the user. For example, according to the scenario of Section 2, $ctx_u(\text{texture}) = \{\text{chewing and swallowing condition}\}$, and $ctx_u(\text{delivery time}) = \{\text{location}\}$. Note that $ctx_u(q) = \emptyset$ if the service delivers fixed value for attribute q at a particular moment regardless of the context of the user invoking the service (e.g. $ctx_u(\text{price}) = \emptyset$).

$ctx_s : Q \rightarrow 2^{C^s}$ is a quality attribute's service context function, which maps each quality attribute $q \in Q$ to the set of service-related context attributes that may affect its values, such that $c_s \in ctx_s(q)$ indicates that context attribute c_s could have an impact on the behaviour of the service so that the same user may observe different values of attribute q under different values of c_s. For example, according to the scenario of Section 2, $ctx_s(\text{taste}) = \{\text{food recipe}\}$.

Finally, $dom : Q \cup C^u \cup C^s \rightarrow 2^{AV}$ is an attribute domain function, which maps each attribute (quality or contextual) to its corresponding domain (the possible values of this attribute), where AV is the set of all possible attribute values (the union of the domains of all attributes). Generally, an attribute (quality or contextual) could be either categorical or numeric (which in turn could be either discrete or continuous). In this paper, however, we assume that $dom(a)$ refers to the discretised domain of

attribute a. While for categorical and discrete attributes, $dom(a)$ corresponds to the original value space of a, for continuous attributes, $dom(a)$ is obtained via applying an appropriate discretisation algorithm on the original value space (a simple example is dividing the original value space into a number of equal ranges, with values $v \in dom(a)$ corresponding to the respective range representatives).

5. CONTEXT-AWARE QOS LEARNING

In our approach, the QoS characteristics of a service are assessed via a learning-enabled agent associated with the service. This agent (which, for example, could be acting on behalf of the service provider or the broker with which the service is registered) exploits a contextually-enriched history of past interactions with the service in order to expose personalised and dynamism-aware QoS information to potential clients. The modelling and learning details of such an agent are presented next.

5.1. Service Observation

For each quality attribute $q \in Q$, the agent receives a stream of service observations, each reporting the outcome encountered for q in a previous interaction with the service, along with the contextual circumstances surrounding this interaction. Formally, a service observation is denoted as $(v_q, \vec{v_u}, \vec{v_s})$, where: $v_q \in dom(q)$ is the value observed for q; $\vec{v_u} = (v_u^1, ..., v_u^m)$ is the vector of observed values for user-side contextual attributes $(c_u^1, ..., c_u^m) \in ctx_u(q)^m$; and $\vec{v_s} = (v_s^1, ..., v_s^k)$ is the vector of observed values for service-side contextual attributes $(c_s^1, ..., c_s^k) \in ctx_s(q)^k$.

5.2. Agent Configuration

The main idea behind our approach is that, for a particular quality attribute, the agent maintains a set of learned *value models*, each corresponding to a different behaviour of the service with respect to this attribute (as outlined before, a service may behave differently as a result of changes in its circumstances, i.e. changes in the service-side context). When previously-encountered service circumstances reoccur, older observations of the service collected under such circumstances become relevant again, and the agent can reuse the respective historical value models (learned from these observations) to make future quality value predictions. Such reuse of previously learned value models facilitates a faster adaptation to a behavioural change of the service (as opposed to re-learning the behaviour from new interactions), and consequently improves the accuracy of quality predictions (see Section 6).

Based on this, the configuration of a service agent, for each quality attribute $q \in Q$, at a particular time step $t \in T$, is a tuple (Ω, wt), where: Ω is the model library of the agent at time step t, containing the set of value models learned for the quality attribute; and $wt : \Omega \rightarrow [0, 1]$ is the value model weighting function at time step t, assigning to each model $\omega \in \Omega$, a weighting factor that reflects its contribution for predicting the attribute's value at time step t. In particular, weights wt are utilised to combine the outputs of models $\omega \in \Omega$, via a weighted average, to produce the final output (value prediction) for the quality attribute at time step t.

5.2.1. *Value Model.* Each model $\omega \in \Omega$ is of the form $q : \psi \leftarrow M$. Here, $q \in Q$ is the quality attribute the value of which the model is trying to predict. Precondition ψ identifies the service-side contextual circumstances under which the

model is valid. Specifically, ψ is a logical formula in disjunctive normal form (DNF) restricting the values of contextual attributes $c_s \in ctx_s(q)$. Each literal in this formula is of the form $cval(c_s) \in V$, with function $cval(c_s)$ denoting the value of context attribute $c_s \in ctx_s(q)$, and $V \subset dom(c_s)$ is a set of values ($V \neq \emptyset$).

Body M is the actual value model, enabling personalised predictions for quality attribute q under condition ψ. In particular, given the different user-side contextual attributes affecting quality attribute q, $(c_u^1, ..., c_u^m) \in ctx_u(q)^m$, value model M corresponds to the underlying function $qval$ between the values of these attributes, $\{\vec{v_u}\} = \{(v_u^1, ..., v_u^m) \mid v_u^1 \in dom(c_u^1), ..., v_u^m \in dom(c_u^m)\}$, and the corresponding value of q. That is, $qval(v_q, \vec{v_u}) \in [0, 1]$ is probability of observing value v_q for attribute q given user's contextual values $\vec{v_u}$, such that $\sum_{v_q \in dom(q)} qval(v_q, \vec{v_u}) = 1$.

Both the precondition ψ and the body M of a model ω are not static, and may evolve over time as the agent's learning progresses. For the former, this occurs as more information regarding the relationship between service-side contextual attributes and service behavioural changes becomes available, while the later is updated to increase the accuracy of function $qval$ with more incoming data. Note that we do not restrict our approach to any specific algorithm for learning function $qval$ (for which several alternatives exist in the literature), and henceforth simply refer to such a value classification algorithm as $classify_{q,u}$. However, we will give an example implementation, utilising a Naïve Bayesian classifier, in Section 6.1.

[Figure 4 about here.]

5.2.2. *Handling Quality Attribute Dependencies.* In some cases, the value of a quality attribute might also be dependent on the values of other quality attributes.

For example, a better taste and a better presentation of a meal might require a longer preparation time, and thus a slower delivery. To incorporate such dependencies, chaining of prediction models can be applied, where the value predicted for one quality attribute is used as an additional input to the prediction model of another attribute (Jesse et al., 2011). Clearly, such chaining would entail that the quality value prediction function $qval$ takes extra input. For example, consider the directed acyclic graph of Figure 4 capturing dependencies among four quality attributes, where edge (q_i, q_j) indicates that the value of attribute q_j is dependent on that of attribute q_i. Denote the predicted value for a quality attribute q_i by $\overline{v_{q_i}}$, and assume (for simplicity of notation) that the user context vector $\vec{v_u}$ is similar for all quality attributes. The chain of predictions can thus be conducted according to the topological order $q_1,\ q_2,\ q_3,\ q_4$, with the inputs of function $qval$ being given as $qval(v_{q_1}, \vec{v_u})$, $qval(v_{q_2}, \vec{v_u}, \overline{v_{q_1}})$, $qval(v_{q_3}, \vec{v_u}, \overline{v_{q_1}})$, and $qval(v_{q_4}, \vec{v_u}, \overline{v_{q_2}}, \overline{v_{q_3}})$, respectively. We do not consider the issue of dependency further, and in the rest of the paper we refer to function $qval$ as solely a function of $\vec{v_u}$, i.e. $qval(v_q, \vec{v_u})$.

5.3. Agent Learning Algorithm

As the agent receives new service observations regarding a quality attribute, it adjusts its respective configuration as detailed in the following sections.

5.3.1. *Model Weight Adjustment.* Inspired by q-learning (Bowling and Veloso, 2001), the adjustment of value model weights wt, for each prediction round, is performed as follows. Given a new service observation at time step t, $(v_q, \vec{v_u}, \vec{v_s})$, and the current agent configuration (Ω^{t-1}, wt^{t-1}), the agent assesses the similarities

between immediate service-side circumstances ψ^{imd} (the conjunction of observed values $\vec{v_s}$), and the contextual preconditions ψ^{t-1} of each model $\omega \in \Omega^{t-1}$, with such similarities being denoted as $sim(\psi^{imd}, \psi^{t-1})$ (see Definition 1 of Section 5.3.3). Based on this, the currently maintained model weights wt^{t-1} are improved according to a learning rate δ_{wt} to reflect these immediate similarities, as:

$$\forall \omega \in \Omega^{t-1},\ wt^t(\omega) = (1 - \delta_{wt}) wt^{t-1}(\omega) + \delta_{wt} sim(\psi^{imd}, \psi^{t-1}) \tag{1}$$

Here, the choice of the learning factor δ_{wt} governs the *adaptivity* property of the weights. Specifically, as δ_{wt} tends to unity, the weight estimation function becomes a greedy function, removing the impact of all previous similarities with the service's contextual circumstances up to step $t - 1$, and accounting only for the latest observation (which facilitates a faster reaction to a change). In contrast, lowering the value of δ_{wt} decreases responsiveness to new contextual data, resulting in a slower adaptation to a change (but improves robustness to noise).

5.3.2. *Model Library Adjustment.* Following the adjustment of model weights, the agent decides on whether a new model need to be added to the library by distinguishing the following two cases.

Case 1. There exists at least one model in the library with circumstances sufficiently similar to recent circumstances, i.e. $\exists \omega \in \Omega^{t-1},\ wt^t(\omega) \geq thrsh$ (where $thrsh$ is a predefined threshold). In this case, no significant behavioural drift is assumed, and observation $(v_q, \vec{v_u})$ is simply used to update the body M of the model with the highest weight wt^t (with the rest of the model library being unchanged): $M^t = classify_{q,u}(M^{t-1}, (v_q, \vec{v_u}))$, where $classify_{q,u}$ is the classification algorithm per value model (as outlined in Section 5.2).

Case 2. There is no model in the library with circumstances sufficiently similar to recent circumstances, i.e. $\forall \omega \in \Omega^{t-1},\ wt^t(\omega) < thrsh$. In this case, the agent suspects a significant change in service behaviour, and sets up a new model ω^n for the attribute, which is added to the model library of the agent: $\Omega^t = \Omega^{t-1} \cup \{\omega^n\}$. The contextual precondition ψ^n of this model is the conjunction of values $\vec{v_s}$, its body M^n is built incrementally from the new incoming observations starting from the current observation $(v_q, \vec{v_u})$, while its weight $wt(\omega^n)$ is set to 1 (due to exactly matching current circumstances).

After the new model ω^n is stabilised (i.e. after $stabilitySize$ incoming observations under similar circumstances), it is compared against the other existing models in the agent's library to verify whether it is actually reflecting a new service behaviour for the attribute (see Section 5.3.4 for details of such comparison). If a similar model ω^{sim} exists in the library, model ω^n is discarded (i.e. removed from the library) to eliminate redundancy, while the contextual precondition ψ^{sim} of model ω^{sim} is generalised to subsume condition ψ^n (see Definition 2 of Section 5.3.3). Note that, whilst the redundant model is discarded, the data reflected in this model (the observations used to build this model) can be used to further adjust the body of model ω^{sim} (if the latter is not stabilised).

Details of contextual operations and model similarity calculation are presented next. Note that one could argue that service-side context could also be included as additional input for function $qval$. Yet, service-side context tends to be fixed over longer periods, during which it does not have any prediction ability. Hence, its inclusion as input into the value model's learning function would only increase the problem dimensionality, creating unnecessary noise (Gomes et al., 2011).

5.3.3. *Contextual Operations.* We define the following operations on contextual conditions (which correspond to restrictions on service-side contextual circumstances, represented in DNF).

Definition 1. Contextual Similarity, $sim(\psi_1, \psi_2)$

Consider two contextual conditions ψ_1 and ψ_2 in DNF. The similarity between these conditions, $sim(\psi_1, \psi_1)$, is estimated as follows:

$$sim(\psi_1, \psi_1) = \max_{\substack{cls_1 \in clauses(\psi_1) \\ cls_2 \in clauses(\psi_2)}} clsSim(cls_1, cls_2)$$

where $clauses(\psi)$ are the conjunctive clauses of condition ψ, and $clsSim(cls_1, cls_2)$ is the similarity between two conjunctive clauses, given as:

$$clsSim(cls_1, cls_2) = 1 - dist(\vec{cls}_1, \vec{cls}_2)$$

Here, $dist(\vec{cls}_1, \vec{cls}_2) \in [0, 1]$ is a distance measure between value vectors $\vec{cls}_1$ and $\vec{cls}_2$, corresponding to conjunctive clauses cls_1 and cls_2, respectively. An example implementation of such a distance measure is presented in Section 6.4.

Definition 2. Generalisation Operator, $\cup^{DNF}$

Consider two contextual conditions ψ_1 and ψ_2 in DNF. The generalisation of condition ψ_1, ψ_1^g, so that it subsumes condition ψ_2, i.e. $\psi_1^g = \psi_1 \cup^{DNF} \psi_2$, is given as follows:

$$clauses(\psi_1^g) = clauses(\psi_1) \cup clauses(\psi_2)$$

where $clauses(\psi)$ are the conjunctive clauses of condition ψ.

5.3.4. *Model Similarity.* For the purpose of comparing two value models, ω and ω', of quality attribute q, we utilise the *conceptual equivalence measure* proposed

by Yang et al. (2006). In particular, the equivalence degree between ω and ω' corresponds to the sum of the similarity scores between these models over a window of $stabilitySize$ records of user-side contextual samples $\{\vec{v_u}\}$. For each record, the similarity score between ω and ω' is set to 1 if both models predict the same output for attribute q given the record's value vector $\vec{v_u}$ as input; otherwise, the similarity score is set to -1. If the overall estimated equivalence degree (devided by $stabilitySize$) is above a pre-defined threshold, models ω and ω' are considered similar.

Note that, when comparing a new model ω^n to other historical models in the agent's library, if several historical models are found equivalent to model ω^n, the one with the highest equivalence degree is chosen.

6. EXPERIMENTS AND RESULTS

In this section, we present an empirical evaluation of the proposed QoS learning framework, focusing on its performance in terms of producing accurate quality value predictions for services in dynamic and user-dependent settings. For simplicity, we only show the results from the perspective of one service and one quality attribute (other attributes and services exhibit similar trends).

An experiment run consists of a number of learning episodes (or cycles) of the service agent. Each of such cycles involves the following three steps. (1) *Observe*: the service delivers particular value v_q for quality attribute q under user's context $\vec{v_u}$ and service's context $\vec{v_s}$, which is observed by the service agent. (2) *Learn*: the service agent utilises this new service observation $(v_q, \vec{v_u}, \vec{v_s})$ to update its current

configuration (as discussed in Section 5.3). (3) *Predict*: the agent uses the adjusted configuration to predict the expected quality value for the next user (i.e. under next user's context $\vec{v_u}'$). In particular, the value predicted for the attribute is the one with the maximum expected probability, where the expected probability $qval^E(v_q, \vec{v_u}')$ of a quality value v_q under user's context $\vec{v_u}'$ is a wighted mean of the probabilities produced by each model $\omega_i \in \Omega$, as follows:

$$qval^E(v_q, \vec{v_u}') = \frac{\sum_{\omega_i \in \Omega} wt(\omega_i) \times qval^{\omega_i}(v_q, \vec{v_u}')}{\sum_{\omega_i \in \Omega} wt(\omega_i)} \quad (2)$$

The actually perceived quality value by this user is observed by the service agent in the next learning episode. All the results are averaged over 100 runs. Other experimental settings are detailed in Table 5.

[Table 5 about here.]

Further details regarding the experimental setup are presented in Sections 6.1 to 6.4, followed by experimental results (Sections 6.5 and 6.6).

6.1. Value Model Implementation

For the purpose of implementing the body M of each model $\omega \in \Omega$, we use the Naïve Bayesian classifier, a statistical classifier based on the Bayes's theorem (Widmer, 1997). In particular, given a quality attribute q and a corresponding observed user's context sample $\vec{v_u} = (v_u^1, ..., v_u^m)$, the probability of attribute q taking on value v_q given evidence $\vec{v_u}$ is the the posterior probability $p(v_q|\vec{v_u})$, computed as follows:

$$p(v_q|\vec{v_u}) = \frac{p(v_q) \times p(\vec{v_u}|v_q)}{p(\vec{v_u})} \quad (3)$$

Here: $p(v_q)$ is the prior probability of value v_q; $p(\vec{v_u})$ is the prior probability of sample $\vec{v_u}$ (this is the same for all the values of q and thus could be omitted);

and $p(\vec{v_u}|v_q)$ is the posterior probability of sample $\vec{v_u}$ conditioned on value v_q. To simplify the computation cost of $p(\vec{v_u}|v_q)$, independence is usually assumed among the attributes of the sample, leading to:

$$p(\vec{v_u}, v_q) = \prod_{i=1}^{m} p(v_u^i|v_q)$$

The estimation of probabilities $p(v_q)$ and $p(v_u^i|v_q)$ can be easily achieved via maintaining corresponding value counts (without the need to store sample records). The incremental learning function, $classify_{q,u}$, thus corresponds to the update of these counts after each new service observation $(v_q, \vec{v_u})$.

Naïve Bayesian classifier is both computationally and memory efficient, making it suitable for the purpose of quality value prediction (which is conducted at run time, while the service is in operation).

6.2. Simulation Framework

We conduct our experiments on a synthetic dataset, allowing us to control the quality values and their changes, thus facilitating evaluation under different settings. In particular, the generation of service quality data in our simulation is based on the data generation framework proposed by Narasimhamurthy and Kuncheva (2007) for classification problems, as detailed in Section 6.2.1. Given the space of possible user context $\{\vec{v_u}\}$, and the space of possible values $\{v_q\}$ for a quality attribute q, the behaviour of a service with respect to attribute q (which we interchangeably refer to as the *data generation source* for q) is characterised by posterior probabilities $p(v_q|\vec{v_u})$, which in turn are determined by prior probabilities $p(v_q)$ and conditional probabilities $p(\vec{v_u}|v_q)$ (see Equation 3).

6.2.1. *Change Model.* As stated earlier, a service's behaviour for an attribute q may experience changes over time. In our simulation, such changes correspond to changes in the data generation source for q, i.e. changes in $p(v_q)$ and/or $p(\vec{v_u}|v_q)$. Generally, drifts may follow various patterns. A drift might occur abruptly, by suddenly switching from one data generation source to another at some time step. Examples of such a drift include a significant degradation in a service's availability due to an unexpected network problem, or modification of service characteristics due to an implementation change. Alternatively, a drift may happen gradually, with data generation exhibiting smaller differences over a longer time period. Examples of such a drift include a slow deterioration of a hardware service performance. Two types of gradual drift can be distinguished. The first type is when a data generation source disappears gradually while another one takes over, i.e. only a few data points are initially generated from the latter source, before it takes over eventually. Another type of gradual drift, which we refer to as incremental drift (Gama et al., 2014), is when there are consecutive intermediary sources between the initial and final data generation sources, i.e. there is a distinct data generation source at each time step t, with the source at time step t differing only slightly from that at time step $t-1$.

To simulate such drift patterns, a number of data generation sources are assumed $S_1, S_2, ..., S_n$. Each source S_i has a particular influence $infl(S_i, t)$ at time step t. Based on this, data at time step t is generated according to the following probabilities:

$$p(v_q) = \sum_{i=1}^{n} infl(S_i, t) * p_i(v_q)$$

$$p(\vec{v_u}|v_q) = \sum_{i=1}^{n} infl(S_i, t) * p_i(\vec{v_u}|v_q)$$

where $p_i(v_q)$ and $p_i(\vec{v_u}|v_q)$ are the prior and conditional probabilities associated with source S_i.

For abrupt and incremental changes, only one source is active at a particular time, while a mixture of sources may be applied in the case of gradual changes. For example, given two generation sources S_1 and S_2, an abrupt (or incremental) change from S_1 to S_2 would correspond to changing the influence vector $\langle infl(S_1,t), infl(S_2,t)\rangle$ from $\langle 1,0\rangle$ to $\langle 0,1\rangle$, whilst a respective gradual change would correspond to gradual influence changes, examples of which are:

$$\langle 1,0\rangle \rightarrow ... \rightarrow \langle 0.75,0.25\rangle \rightarrow ... \rightarrow \langle 0.25,0.75\rangle \rightarrow ... \rightarrow \langle 0,1\rangle$$

6.2.2. *Service Execution Model.* The dataset generated in our simulation is inspired by STAGGER concepts (Schlimmer and Granger, 1986; Widmer, 1997), but is adapted to suit our example food provision scenario, considering food taste as the quality attribute under prediction (other attributes, such as delivery time, can be represented similarly, but are not considered in the simulation).

Food taste can be influenced by a number of factors, examples of which include:

- Age: taste discrimination tends to decrease with age, with taste thresholds for sweetness, saltiness, and sourness in the elderly being different from those in the young.
- Health Condition: a number of diseases may affect taste sensitivity, including obesity, anorexia, etc.
- Culture: cultural influences may lead to differences in food habits, with individuals from a particular country favouring particular traditions of food preparation.

Based on this, we assume the following in our simulation:

- five possible outcomes for quality attribute $taste$, $dom(taste) = \{1, 2, 3, 4, 5\}$, corresponding to the number of stars assigned by the user;
- three user-side context attributes, Age, $HealthCondition$, and $Culture$, affecting $taste$, $\vec{v_u} = \langle Age, HealthCondition, Culture \rangle$, each with three possible values,

$$dom(Age) = \{age1, age2, age3\}$$

$$dom(HealthCondition) = \{cnd1, cnd2, cnd3\}$$

$$dom(Culture) = \{cul1, cul2, cul3\}$$

- one service-side context attribute, $Recipe$, also with three possible values,

$$dom(Recipe) = \{recipe1, recipe2, recipe3\}$$

- and finally three different service behaviours regarding attribute $Taste$, as defined in Figure 5. Each behaviour corresponds to a distinct data generating source, as characterised in Figure 6.

[Figure 5 about here.]

[Figure 6 about here.]

The above sources are adopted for the purpose of simulating abrupt and gradual changes. Now, to facilitate the simulation of incremental changes, we assume that the distributions of input features (user contextual features) per each class label, i.e. conditional distributions $p(c_u^i|v_q)$, are Gaussian, with their respective means being denoted as $\mu(c_u^i, v_q)$ (the mean corresponding to the distribution of feature c_u^i for class label v_q). All possible outcomes are assigned equal prior probabilities, $p(v_q)$, in this case. Distribution means are initiated at the beginning of the simulation, and slightly repositioned at each time step (with a change step δ_u) to simulate slowly

changing data generation sources:

$$\mu_t(c_u^i, v_q) = \mu_{t-1}(c_u^i, v_q) + \delta_u$$

For all change types, a service-related context attribute is assumed to follow a normal distribution, with its mean being repositioned either abruptly (with a magnetite of change Δ_s) at each change point in the case of abrupt changes, or slowly (with a magnetite of change $\delta_s = \frac{\Delta_s}{|T|}$) at each time step in the case of incremental and gradual changes, with T being the set of all time steps (or learning episodes).

6.3. Evaluation Strategies and Measure

Throughout the evaluation, we refer to the following quality value learning strategies. Strategy *MML*, our proposed multi-model learning approach. Strategy *SSL*, a simple summary-based learning approach, which predicts the quality value v_q with the highest prior probability, $p(v_q)$, based on all the observations so far and ignoring the user's context. Finally, strategy *SWL_*w, a sliding window based learning approach, a well known way in the literature of adapting to potential changes in incoming data (Gama et al., 2014). It utilises the Naïve Bayesian classifier presented in Section 6.1 as its main model, but maintains a fixed window of the latest w observations, based upon which the model is updated at each time step (this strategy accounts for the user's context, but ignores the service's context). By *SWL_*all, we refer to accounting for *all* the data observed so far.

The performance of each learning strategy is evaluated by assessing its prediction accuracy at each time step, calculated as the success rate (i.e. $\frac{\text{number of successful predictions}}{\text{total number of predictions}}$) over the last o observations (o is set to 20 in our experiments).

6.4. Distance Measure

In the simulation, we consider one service-related contextual attribute c_s, with the distance, $dist$, between two values of this attribute, v_s^1 and v_s^2, being estimated as follows:

$$dist(v_s^1, v_s^2) = \begin{cases} \frac{|v_s^1 - v_s^2|}{CDT}, \text{ if } |v_s^1 - v_s^2| <= CDT \\ 1, \text{ otherwise} \end{cases}$$

Here, CDT is the context dissimilarity threshold, beyond which the two values are considered totally dissimilar, i.e. the observations of service behaviour collected under one of these values are no longer relevant under the other.

Based on this distance measure, the classifier weight threshold, $thrsh$ (see Section 5.3.2), which triggers building a new classifier when the weights associated with existing classifiers fail to meet this threshold, can be expressed as follows:

$$thrsh = 1 - \frac{NCT}{CDT}$$

Here, NCT (s.t. $NCT \leq CDT$) is the new classifier distance threshold, denoting the maximum distance allowed between current and a previous service-related context, before the need for initiating a new classifier. Note that, when $CDT > NCT$, an older classifier may still carry some weight on the current prediction, even after a new one is initiated. Further analysis of the effect of different values of CDT and NCT is provided in the following sections.

6.5. Results in Static Environment

Our goal here is to study the importance of user context awareness for producing more accurate quality value predictions (tailored towards the user's particular situ-

ation). For this purpose, we assume a static service behaviour regarding attribute q (e.g. the service always conforms to *behaviour 2*), and compare our learning strategy, *MML*, against the simple summary one, *SSL*. To simulate situations of imperfect user's contextual knowledge, we run our approach at different levels of noise, $\eta\%$ (i.e. the true value of attribute q in a service observation is replaced with a randomly generated one from its domain according to probability $\eta\%$).

The results in Figure 7 demonstrate that *MML* achieves an accuracy of over 80% (at 0% noise) after only 30 cycles, as opposed to *SSL* where the accuracy fluctuates around 50% for the entire run. It is also evident that even with high noise levels (e.g. 30% noise), *MML* still leads to significant improvements over *SSL*, indicating the importance of accounting for contextual evidence, even if imperfect.

[Figure 7 about here.]

6.6. Results in Dynamic Environment

The goal here is to study the adaptivity of the proposed approach in dynamic environments, where the behaviour of the service changes over time. Different types of dynamic environments are analysed next.

6.6.1. *Incremental Changes.* Figure 8 shows the results of the considered strategies in the case of incremental changes, where the means of data distributions representing user and service contextual features are slightly repositioned at each time step.

[Figure 8 about here.]

As expected, the performance of *SWL_all* deteriorates with time as older ob-

servations become less relevant. Here, better prediction accuracy is obtained when the outdated data is gradually forgotten, favouring more recent observations, with *SWL*_100 achieving good results. With appropriate settings of parameters CDT and NCT, the proposed multi-model strategy is also able to gradually discount the effect of older irrelevant data, managing to approximate the performance of *SWL*_100 when $CDT = 0.2 \times \Delta_s,\ NCT = 0.1 \times \Delta_s$, where Δ_s is the overall magnitude of change for the service-side context attribute within a run.

[Figure 9 about here.]

[Figure 10 about here.]

Figures 9 and 10 further analyse the effect of different settings for parameters CDT and NCT in this case. In Figure 9, parameter NCT is fixed at $0.1 \times \Delta_s$, resulting in 0.1 of the total samples contributing to each individual classifier, while varying parameter CDT. Here, setting CDT to higher values expands the window of past samples that could have an effect on the current prediction. In other words, whilst the most recent classifier still carries the highest weight, more past classifiers would be allowed to also contribute towards the prediction at hand (see Figure 11), intensifying the effect of irrelevant data and thus reducing accuracy. On the other hand, when $CDT = NCT$, only a single classifier (the most recent one) is active at any time step. Although this reduces the effect of the less relevant data captured by the older classifiers, it suffers from accuracy drops at classifier change points, until the new classifier is stabilised, due to the lack of sufficient samples. A smoother transition, and thus better accuracy, is obtained when the older classifier is allowed to contribute towards the prediction, but with lower weight, while the new classifier

is being build, which is achieved via setting CDT to $0.2 \times \Delta_s$ (i.e. two classifiers are active at each time step as outlined in Figure 11).

[Figure 11 about here.]

Now, in Figure 10, CDT is always set to $0.3 \times \Delta_s$, while parameter NCT is the one varied. Here, the maximum number of past samples that may contribute towards the prediction remains the same, but depending on the value of NCT, such samples are either broken down into smaller chunks (i.e. larger number of classifiers) for lower values of NCT, or larger chunks (i.e. smaller number of classifiers) for higher values of NCT, in ascending order of importance. In particular, according to our model, the importance of a classifier is governed by the contextual characteristics of its *initial samples* (the similarity between these characteristics and the current context). As a result, when classifiers are larger (for higher values of NCT, e.g. $NCT = 0.3 \times \Delta_s$), a newly built classifier carries a much higher weight than the older ones. This leads to accuracy drops each time a new classifier is initiated, followed by a boost in performance after the new classifier is stabilised, before the accuracy starts dropping again as the window of samples reflected by the classifier increases capturing older irrelevant samples. In contrast, when classifiers are smaller (for lower values of NCT, e.g. $NCT = 0.1 \times \Delta_s$), the older classifier still holds a considerable importance when the new one is built, resulting in a smoother transition, but lower accuracy peaks due to the higher impact of older samples. Furthermore, as can be seen, values as low as $NCT = 0.03 \times \Delta_s$ leads to a considerable reduction in accuracy due to relying on a large number of unstable classifiers built on insufficient number of samples.

[Figure 12 about here.]

6.6.2. *Gradual Changes.* Figure 12 shows the results of the considered strategies in the case of gradual changes, where a data source gradually disappears to be replaced by another source each 500 time steps, following sequence $S_1 - S_2 - S_3$ (see Figure 6 for the definition of sources S_i). We obtain similar observations to the case of incremental changes in terms of ranking between strategies, i.e. better results are achieved when older data is gradually forgotten. Note, however, that we encounter here an increase in the performance of all classifiers between time steps 300 and 700. This is due to the pattern of changes followed, where data source 2 remains the source dominating ($infl(S_2, t) > 0.5$) for this entire period, which is longer than the domination period of the other sources, resulting in a milder change within this period.

6.6.3. *Abrupt Changes.* Finally, we compare the performance of the adaptation strategies in the case of abrupt changes, where a data source suddenly changes to another source following the behaviour sequence $S_1 - S_2 - S_3 - S_1 - S_2 - S_3$, with each particular behaviour being fixed for 300 episodes.

[Figure 13 about here.]

Figure 13 shows the results in the case of encountered *new behaviour* (i.e. the case of changes during the first 900 episodes). As can be seen, *SWL_all* suffers from poor performance, especially after a change occurrence, where the learned model mostly reflects irrelevant observations. It is also evident that, in fixed windowing strategies, increasing the window size results in a slower reactivity to a change

since older irrelevant observations take longer to be forgotten. Smaller windows, on the other hand, achieve faster adaptation, but affect the prediction accuracy due to depending on insufficient number of observations. In contrast, via accounting for service-side contextual clues (with appropriate settings of parameters CDT and NCT), *MML* is able to *detect* a drift occurrence, and utilise this to capture only relevant observations for learning the new behaviour (i.e. acting similarly to a dynamically adjusting window), thus outperforming other strategies.

In order for *MML* to be able to *detect* a drift occurrence in this case, parameter NCT should be set to a value $NCT \leq \Delta_s$ (where Δ_s is the magnetite of service-related context change between behaviours). Moreover, since the change is abrupt, data preceding the change point is no longer relevant and should be forgotten abruptly, which can be achieved by setting parameter CDT to: $NCT \leq CDT \leq \Delta_s$, allowing only one classifier to be active at any moment. As the value of CDT increases beyond Δ_s, so does the weight of the older, no longer relevant classifier, resulting in the degradation of prediction accuracy as illustrated in Figure 14.

[Figure 14 about here.]

[Figure 15 about here.]

As for the case of encountered *recurring behaviour* (i.e. the case of changes during the second 900 episodes), and unlike the other strategies, *MML* always maintains high accuracy (see Figure 15), eliminating the period of performance degradation after a change point due to reusing an already existing stable model (built from old, but relevant again, observations).

[Figure 16 about here.]

[Figure 17 about here.]

6.6.4. *Effect of Noise.* The effect of imperfect (e.g. incomplete) service-side context knowledge on the performance of *MML* is studied in Figure 16, where context attribute c_s is subjected to various levels of noise $\eta\%$, in an abruptly changing environment (other changing environments exhibit similar trends). Clearly, the presence of noise has a negative impact on accuracy. Here, the choice of classifier weight learning factor δ_{wt} plays a role in robustness to noise, with lower values of δ_{wt} (i.e. slower learning), e.g. $\delta_{wt} = 0.1$, achieving better results due to decreasing responsiveness to noisy data, as shown in Figure 17 where the context attribute is subjected to 30% noise.

7. RELATED WORK

The QoS properties of services are important criteria upon which services are discovered, selected, and composed into complex applications (Zeng et al., 2004; Barakat et al., 2011; Yan and Chen, 2015). Accurate estimation of such properties, typically from prior observations of service behaviour, has thus received much attention.

7.1. QoS Prediction

Many approaches have been proposed in the context of QoS prediction. For example, Aschoff and Zisman (2011) model the response time of a service as a random variable, changing as a result of various factors related to the network and system resources (e.g. request queuing time). The exponentially weighted moving

average is utilised for estimating the expected value of this variable at a particular time step, according to historical data. Similarly, time series modelling based on ARIMA (AutoRegressive Integrated Moving Average) has been proposed by Amin et al. (2012) for the purpose of QoS forecasting. Barakat et al. (2014) provide probabilistic, multi-valued quality estimations for services via applying an online learning algorithm that is inspired by Policy Hill-Climbing, based on past user ratings.

Trust and reputation mechanisms have also been considered for the purpose of accurate quality predictions. Trust is defined as an assessment of the likelihood that an individual or organisation will cooperate and fulfil its commitments (Gambetta, 1988). Reputation is complementary to trust, and can be viewed as the public perception of the trustworthiness of a given entity (Jøsang et al., 2007). In a service-oriented system individuals and organisations rely on providers to successfully execute services with an appropriate quality in order to fulfil their own goals, and such reliance implies a degree of risk, as success depends in part upon a third party. Trust and reputation provide an effective way of assessing and managing this risk with respect to the quality attributes of interest. In particular, prior to an interaction with a service, an assessment of its overall trustworthiness (Xu et al., 2007; Malik and Bouguettaya, 2009) or the trustworthiness of each of its QoS dimensions (Maximilien and Singh, 2005) is undertaken, in order to avoid selecting unreliable services that may not honour their promises. Typically, such an assessment is performed by producing reputation scores for the service based on the feedback collected from its users.

Many computation models of trust and reputation have also been developed and applied in a variety of other settings including Grid computing and P2P systems, and more generally in multi-agent systems. See (Griffiths and Chao, 2010; Jøsang et al.,

2007; Sabater and Sierra, 2005) for extensive reviews of the main approaches. Most established trust models, such as ReGreT (Sabater-Mir and Sierra, 2001; Sabater, 2004), FIRE (Huynh et al., 2006), TRAVOS (Teacy et al., 2005) and HABIT (Teacy et al., 2012) use a combination of direct experience and third party experiences as the base for assessing trust and reputation, and use numerical or probabilistic representations for trust (Wang and Singh, 2007). TRAVOS (Teacy et al., 2005) takes a probabilistic approach to assessing trust, with the outcomes of interactions recorded as binary variables from which the expected value of success of future interactions is estimated using a beta probability density function. HABIT (Teacy et al., 2012) also uses a probabilistic approach, creating Bayesian network to support reasoning about reputation. ReGreT (Sabater-Mir and Sierra, 2001; Sabater, 2004) assesses reputation on three aspects: (i) an individual dimension from direct experience, (ii) a social dimension using knowledge of others' experiences and the social structure, and (iii) an ontological dimension that accounts for the different aspects that inform reputation (e.g. delivery, price, quality). FIRE (Huynh et al., 2006) builds on ReGreT through the addition of role-based trust, and certified reputation based on third-party references (Huynh et al., 2006).

7.2. Context Awareness in QoS Prediction

7.2.1. *Time as Context.* Time is the most commonly accounted for context in QoS prediction approaches. In particular, most above approaches mainly rely on the time of service observation to determine the relevance of this observation for the situation at hand. That is, observations in the distant past are considered less relevant than more recent ones. This is achieved by simply discarding older observations

(e.g. Teacy et al. (2005, 2012)), by weighting observations based on their recency (e.g. Huynh et al. (2006); Maximilien and Singh (2005); Xu et al. (2007)), or by introducing a learning factor determining the rate at which past data is forgotten (e.g. Barakat et al. (2014)).

These efforts rely on favouring recent observations to handle changes in the service's behaviour, without accounting for the context under which these observations were collected, thus neglecting important evidence for assessing observation relevance. In contrast, our approach guides its reasoning by such contextual clues in order to achieve more effective adaptation. Hence, it is capable of detecting the behaviour change, as well as reusing old observations in the case of re-appearing behaviour, unlike these efforts.

Moreover, such approaches usually do not function well if there is a lack of past evidence, such as where new providers or services are introduced to the system or where a service under consideration has had little recent use (Burnett et al., 2011). The approach proposed in this paper uses context records that could provide a rich source of information on which to base assessment, enabling a richer level of reasoning via considering the context of the previous interactions.

7.2.2. *Implicit Context Awareness.* To facilitate personalised QoS information for users, a number of approaches utilise collaborative filtering for the purpose of quality value prediction (Shao et al., 2007; Zheng et al., 2009, 2011). In particular, the missing QoS values for a given user are predicted based on the past respective ratings of other *similar* users, with the similarity between two users being determined according to their ratings for the services they commonly invoked previously.

To address the scalability issue underlying accessing the entire database of user-service information in such collaborative filtering approaches, Yu and Huang (2016) propose dividing the group of users into a number of sub-groups based on the user's location. Such division is based on the assumption that the location of a user is likely to affect their perception of a number of QoS information (e.g. response time). This allows seeking user-service data within smaller clusters, thus improving scalability. A similar location-aware collaborative filtering approach is utilised by Tang et al. (2012), seeking the data of users within physical proximity to the target user.

Although collaborative filtering approaches capture the user's context implicitly, they usually suffer from the data sparsity problem where a user's service invocation history is either lacking or insufficient to make similarity conclusions. In contrast, our approach explicitly exploits available user's contextual knowledge, which allows deriving *personalised* quality assessments for the user even in the absence of previous interactions between this user and any service.

7.2.3. *Explicit Context Awareness.* Some approaches have also considered capturing context information explicitly. Like us, Lin et al. (2012) explicitly incorporate the user's contextual attributes as input for the quality value prediction process (which is based on the average of past data). Yet, they do not account for changes in the service's behaviour that are associated with service-side context. A user context aware approach (without service context awareness) is simulated in our experiments via strategies *SWL_*w.

In the context of assessing a provider's reputation, Miles and Griffiths (2015b,a) utilise knowledge of the circumstances under which a service provision occurred.

Specifically, this knowledge is used to scale the ratings available for the provider, assigning higher weights to those collected under circumstances comparable to the current settings. Similarly, Barakat et al. (2015) incorporate delegation information underlying a composite service provision to provide more accurate reputation assessment of the composite service provider. These approaches, however, ignore the user's context and thus, unlike the proposed approach, do not provide personalised predictions, nor consider the issue of subjectivity.

8. CONCLUSION

This paper presented a context-aware QoS learning approach for personalised and adaptive quality value estimations. The learning is conducted via a service agent, which maintains a pool of quality prediction models; each characterising a particular service behaviour, and providing (under this behaviour) personalised value predictions for users. Experimental results demonstrate that the proposed approach achieves high accuracy in different types of changing environments, and a faster adaptation to a change when compared to the commonly adopted time-based learning.

Future work involves exploring hierarchical structures for context information, as well as investigating how the predictive models of other similar services (possibly maintained by other agents) could be exploited by the service agent to enhance quality predictions for a newly available service (i.e. in the absence of prior interaction history with the service).

REFERENCES

AMIN, A., A. COLMAN, and L. GRUNSKE. 2012. An approach to forecasting QoS attributes of web services based on ARIMA and GARCH models. *In* Proc. IEEE Int. Conf. on Web Services, pp. 74–81.

ASCHOFF, R., and A. ZISMAN. 2011. QoS-driven proactive adaptation of service composition. *In* Proc. Int. Conf. on Service-Oriented Computing, pp. 421–435.

BARAKAT, L., J. BARRERA, J. CHARVILL, S. RIVAS, V. SÁNCHEZ, A. TAWEEL, and S. MILES. 2014. Reliability-aware marketplace for food services. *In* eChallenges e-2014, 2014 Conference, IEEE, pp. 1–8.

BARAKAT, L., S. MAHMOUD, S. MILES, A. TAWEEL, and M. LUCK. 2014. An agent-based service marketplace for dynamic and unreliable settings. *In* Proc. Int. Conf. on Service-Oriented Computing, pp. 169–183.

BARAKAT, L., S. MAHMOUD, P. TAYLOR, N. GRIFFITHS, and S. MILES. 2016. Reputation-based provider incentivisation for provenance provision: (extended abstract). *In* Proceedings of the 2016 International Conference on Autonomous Agents & Multiagent Systems, Singapore, May 9-13, 2016, pp. 1429–1430.

BARAKAT, L., S. MILES, L. POERNOMO, and M. LUCK. 2011. Efficient multi-granularity service composition. *In* Proc. 2011 IEEE Int. Conf. on Web Services, pp. 227–234.

BARAKAT, L., P. TAYLOR, N. GRIFFITHS, and S. MILES. 2015. Context-driven assessment of provider reputation in composite provision scenarios. *In* Proceedings of the 13th International Conference on Service Oriented Computing.

BOWLING, M., and M. VELOSO. 2001. Rational and convergent learning in stochastic games. *In* Proceedings of the 17th International Joint Conference on Artificial Intelligence, pp. 1021–1026.

BURNETT, C., T. NORMAN, and K. SYCARA. 2011. Trust decision-making in multi-agent systems. *In* Proceedings of the 22nd International Joint Conference on Artificial Intelligence, pp. 115–120.

CURCIN, V., S. MILES, R. DANGER, Y. CHEN, R. BACHE, and A. TAWEEL. 2014. Implementing interoperable provenance in biomedical research. Future Generation Computer Systems, **34**:1–16.

GAMA, J., I. ŽLIOBAITĖ, A. BIFET, M. PECHENIZKIY, and A. BOUCHACHIA. 2014. A survey on concept drift adaptation. ACM Computing Surveys, **46**:1–37.

GAMBETTA, D. 1988. Can we trust trust? *In* Trust: Making and Breaking Cooperative Relations. *Edited by* D. Gambetta, pp. 213–237.

GOMES, J. B., M.M. GABER, P. A.C. SOUSA, and E. MENASALVAS. 2011. Context-aware collaborative data stream mining in ubiquitous devices. *In* Advances in Intelligent Data Analysis X, pp. 22–33.

GRIFFITHS, N., and K.-M. CHAO editors. 2010. Agent-Based Service-Oriented Computing. Springer.

HUYNH, T.D., N.R. JENNINGS, and N.R. SHADBOLT. 2006. An integrated trust and reputation model for open multi-agent systems. J. of Autonomous Agents and Multi-Agent Systems, **13**(2):119–154.

JESSE, J., B. PFAHRINGER, G. HOLMES, and E. FRANK. 2011. Classifier chains for multi-label classification. Machine Learning, **85**(3):333.

JØSANG, A., R. ISMAIL, and C. BOYD. 2007. A survey of trust and reputation systems for online service provision. Decision Support Systems, **43**(2):618–644.

LIN, D., C. SHI, and T. ISHIDA. 2012. Dynamic service selection based on context-aware QoS. *In* Proc. IEEE Int. Conf. on Services Computing, pp. 641–648.

MALIK, Z., and A. BOUGUETTAYA. 2009. RATEWeb: Reputation assessment for trust establishment among web services. The VLDB Journal, **18**:885–911.

MAXIMILIEN, E.M., and M.P. SINGH. 2005. Agent-based trust model involving multiple qualities. *In* Proc. Int. Joint Conf. on Autonomous Agents and Multiagent Systems, pp. 519–526.

MILES, S., and N. GRIFFITHS. 2015a. Accounting for circumstances in reputation assessment. *In* Proceedings of the 14th International Conference on Autonomous Agents and Multiagent Systems, pp. 1653–1654.

MILES, S., and N. GRIFFITHS. 2015b. Incorporating mitigating circumstances into reputation assessment. *In* Proceedings of the 2nd International Workshop on Multiagent Foundations of Social Computing.

MILES, S., P. GROTH, S. MUNROE, and L. MOREAU. 2011. PrIMe: A methodology for developing provenance-aware applications. ACM Transactions on Software Engineering and Methodology, **20**(3):8:1–8:42.

NARASIMHAMURTHY, A., and L.I. KUNCHEVA. 2007. A framework for generating data to simulate changing environments. *In* Proceedings of the 25th IASTED International Multi-Conference: Artificial Intelligence and Applications, AIAP'07, ACTA Press, Anaheim, CA, USA, pp. 384–389. `http://dl.acm.org/citation.cfm?id=1295303.1295369`.

SABATER, J. 2004. Evaluating the ReGreT system. Applied Artificial Intelligence, **18**(9-10):797–813.

SABATER, J., and C. SIERRA. 2005. Review on computations trust and reputation models. Artificial Intelligence Review, **24**(1):33–60.

SABATER-MIR, J., and C. SIERRA. 2001. Regret: A reputation model in gregarious societies. *In* Proc. of the 4th Workshop on Deception, Fraud and Trust in Agent Societies, pp. 61–69.

SCHLIMMER, J.C., and R.H. GRANGER. 1986. Incremental learning from noisy data. Machine Learning, **1**(3):317–354. ISSN 0885-6125.

SHAO, L., J. ZHANG, Y. WEI, J. ZHAO, B. XIE, and H. MEI. 2007. Personalized QoS prediction for web services via collaborative filtering. *In* Proc. 2007 IEEE Int. Conf. on Web Services, pp. 439–446.

STEPHAN, G., H. PASCAL, and A. ANDREAS. 2007. Knowledge Representation and Ontologies, pp. 51–105.

TANG, M., Y. JIANG, J. LIU, and X. LIU. 2012. Location-aware collaborative filtering for QoS-based service recommendation. *In* Proceedings of the 2012 IEEE 19th International Conference on Web Services, pp. 202–209.

TAWEEL, A., L. BARAKAT, and S. MILES. 2014. A distributed service-based system for homecare self-management. *In* OTM Workshops, pp. 361–366.

TAWEEL, A., L. BARAKAT, S. MILES, T. CIOARA, I. ANGHEL, A.H. TAWIL, and I. SALOMIE. 2016. A service-based system for malnutrition prevention and self-management. Computer Standards & Interfaces, **48**:225–233.

TEACY, W. T. LUKE, MICHAEL LUCK, ALEX ROGERS, and NICHOLAS R. JENNINGS. 2012. An efficient and versatile approach to trust and reputation using hierarchical bayesian modelling. Artificial Intelligence, **193**:149–185.

TEACY, W. T. L., J. PATEL, N. R. JENNINGS, and M. LUCK. 2005. Coping with inaccurate reputation sources: Experimental analysis of a probabilistic trust model. *In* Proc. of the 4th Int. Conf. on Autonomous Agents and Multiagent Systems, pp. 997–1004.

W3C. 2013. PROV model primer. http://www.w3.org/TR/prov-primer/.

WANG, Y., and M.P. SINGH. 2007. Formal trust model for multiagent systems. *In* Proceedings of the 20th International Joint Conference on Artificial Intelligence, pp. 1551–1556.

WIDMER, G. 1997. Tracking context changes through meta-learning. Machine Learning, **27**(3):259–286.

XU, Z., P. MARTIN, W. POWLEY, and F. ZULKERNINE. 2007. Reputation-enhanced QoS-based web services discovery. *In* Proc. IEEE Int. Conf. on Web Services, pp. 249–256.

YAN, Y., and M. CHEN. 2015. Anytime QoS-aware service composition over the GraphPlan. Service Oriented Computing and Applications, **9**(1):1–19.

YANG, Y., X. WU, and X. ZHU. 2006. Mining in anticipation for concept change: Proactive-reactive prediction in data streams. Data Mining and Knowledge Discovery, **13**(3):261–289.

YU, C., and L. HUANG. 2016. A web service QoS prediction approach based on time- and location-aware collaborative filtering. Service Oriented Computing and Applications, **10**(2):135–149.

ZENG, L., B. BENATALLAH, A.H.H. NGU, M. DUMAS, J. KALAGNANAM, and H. CHANG. 2004. QoS-aware middleware for web services composition. IEEE Transactions on Software Engineering, **30**(5):311–327.

ZHENG, Z., H. MA, M.R. LYU, and I. KING. 2009. WSRec: A collaborative filtering based web service recommender system. *In* Proc. IEEE Int. Conf. on Web Services, pp. 437–444.

ZHENG, Z., H. MA, M.R. LYU, and I. KING. 2011. QoS-Aware web service recommendation by collaborative filtering. IEEE Trans. Serv. Comput., **4**(2):140–152.

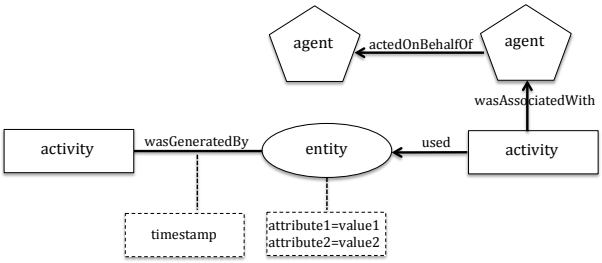

FIGURE 1. PROV graph illustrating the key elements

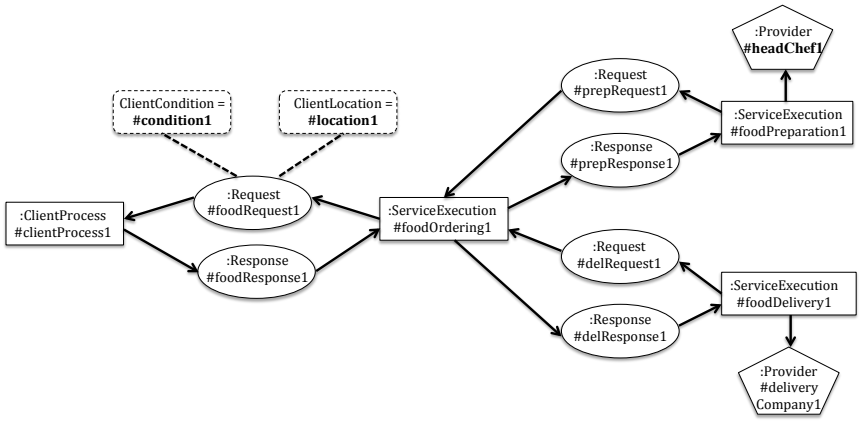

FIGURE 2. PROV graph illustrating the contextual circumstances presented in the motivating scenario

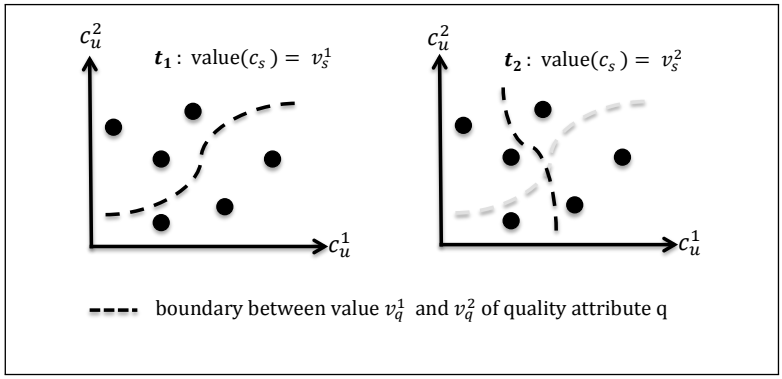

FIGURE 3. Dependency between the values of a quality attribute and the user's and service's contextual features

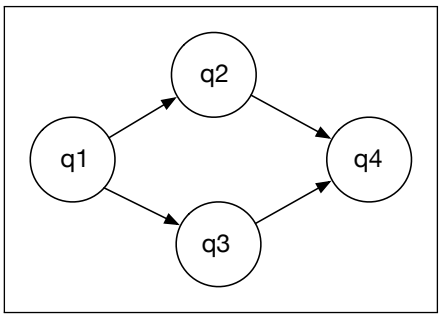

FIGURE 4. Example Dependency Graph among Quality Attributes

(1) $behaviour1$. under recipe 1:

$$\text{ranking of } taste = \begin{cases} \text{1 star} & \text{under } age1 \wedge cnd1 \\ \text{2 star} & \text{otherwise} \end{cases}$$

(2) $behaviour2$. under recipe 2:

$$\text{ranking of } taste = \begin{cases} \text{1 star} & \text{under } cnd2 \vee cul2 \\ \text{3 star} & \text{otherwise} \end{cases}$$

(3) $behaviour3$. under recipe 3:

$$\text{ranking of } taste = \begin{cases} \text{4 star} & \text{under } age2 \vee age3 \\ \text{5 star} & \text{otherwise} \end{cases}$$

FIGURE 5. Definition of service behaviour regarding quality attribute $taste$

(1) **Behaviour 1 (associated with Source S_1)**

$$p_1(1\text{ star}) = 3/27,\ \ p_2(2\text{ star}) = 24/27,\ \ p_3(3\text{ star}) = 0,\ \ p_4(4\text{ star}) = 0,\ \ p_5(5\text{ star}) = 0$$

$$p_1(\vec{v_u}|1\text{ star}) = \begin{cases} 1/3, \text{ if } \vec{v_u} = \langle age1, cnd1, * \rangle \\ 0, \text{ otherwise} \end{cases}$$

$$p_1(\vec{v_u}|2\text{ star}) = \begin{cases} 0, \text{ if } \vec{v_u} = \langle age1, cnd1, * \rangle \\ 1/24, \text{ otherwise} \end{cases}$$

(2) **Behaviour 2 (associated with Source S_2)**

$$p_2(1\text{ star}) = 15/27,\ \ p_2(2\text{ star}) = 0,\ \ p_2(3\text{ star}) = 12/27,\ \ p_2(4\text{ star}) = 0,\ \ p_2(5\text{ star}) = 0$$

$$p_2(\vec{v_u}|1\text{ star}) = \begin{cases} 1/15, \text{ if } \vec{v_u} = \langle *, cnd2, * \rangle \vee \langle *, *, cul2 \rangle \\ 0, \text{ otherwise} \end{cases}$$

$$p_2(\vec{v_u}|3\text{ star}) = \begin{cases} 0, \text{ if } \vec{v_u} = \langle *, cnd2, * \rangle \vee \langle *, *, cul2 \rangle \\ 1/12, \text{ otherwise} \end{cases}$$

(3) **Behaviour 3 (associated with Source S_3)**

$$p_3(1\text{ star}) = 0,\ \ p_3(2\text{ star}) = 0,\ \ p_3(3\text{ star}) = 0,\ \ p_3(4\text{ star}) = 18/27,\ \ p_3(5\text{ star}) = 9/27$$

$$p_3(\vec{v_u}|4\text{ star}) = \begin{cases} 1/18, \text{ if } \vec{v_u} = \langle age2, *, * \rangle \vee \langle age3, *, * \rangle \\ 0, \text{ otherwise} \end{cases}$$

$$p_3(\vec{v_u}|5\text{ star}) = \begin{cases} 0, \text{ if } \vec{v_u} = \langle age2, *, * \rangle \vee \langle age3, *, * \rangle \\ 1/9, \text{ otherwise} \end{cases}$$

FIGURE 6. Data generation sources corresponding to the behaviours of Figure 5

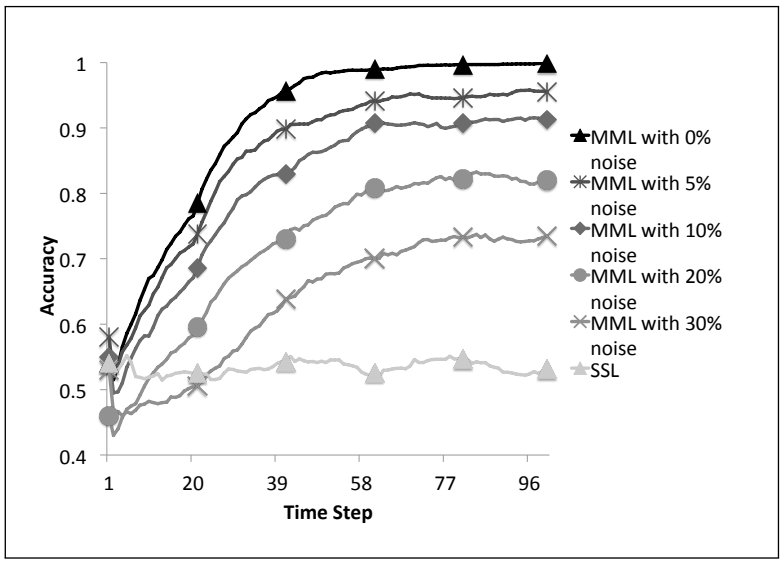

FIGURE 7. Effect of Personalisation

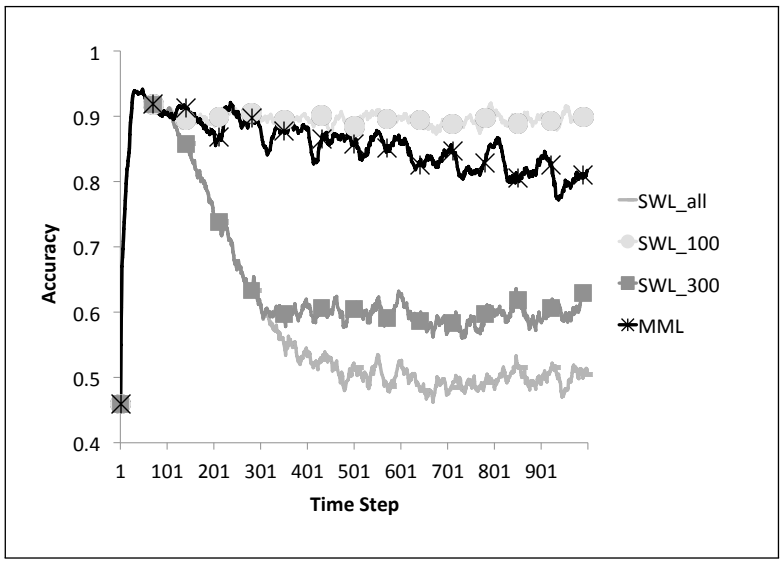

FIGURE 8. Incremental Changes

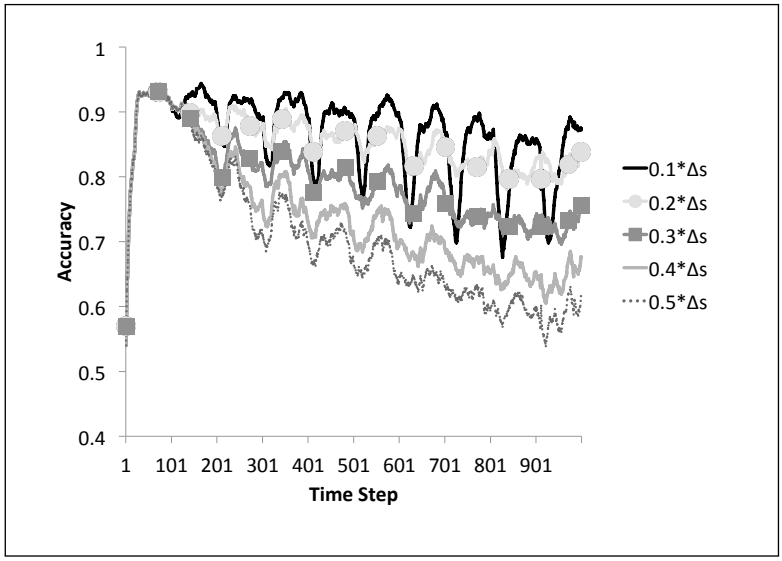

FIGURE 9. Incremental Changes - Effect of CDT

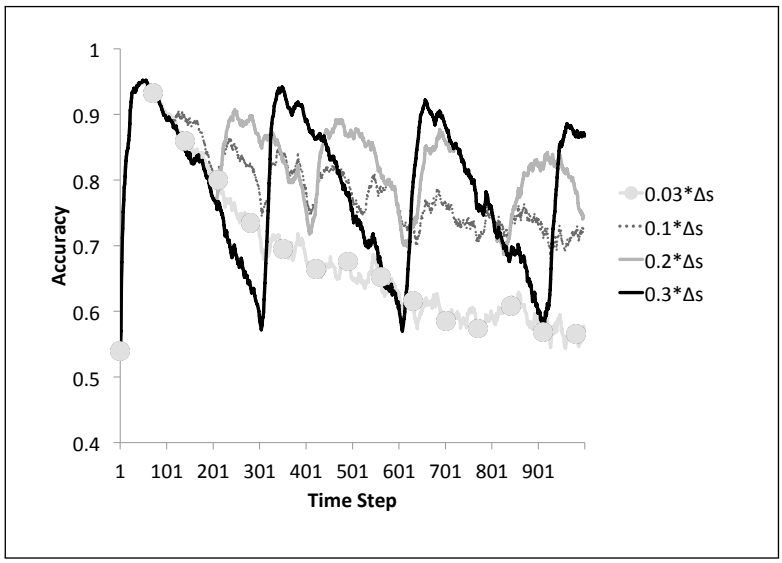

FIGURE 10. Incremental Changes - Effect of NCT

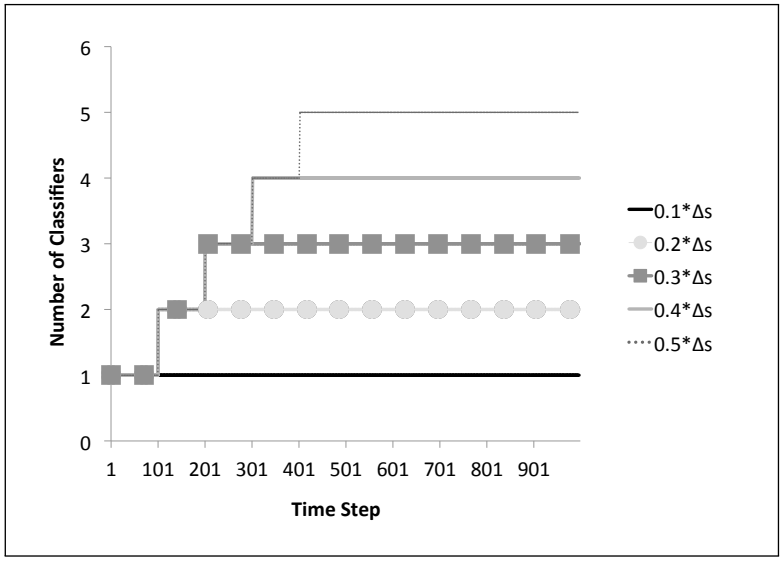

FIGURE 11. Incremental Changes - Effect of CDT - Number of Active Classifiers

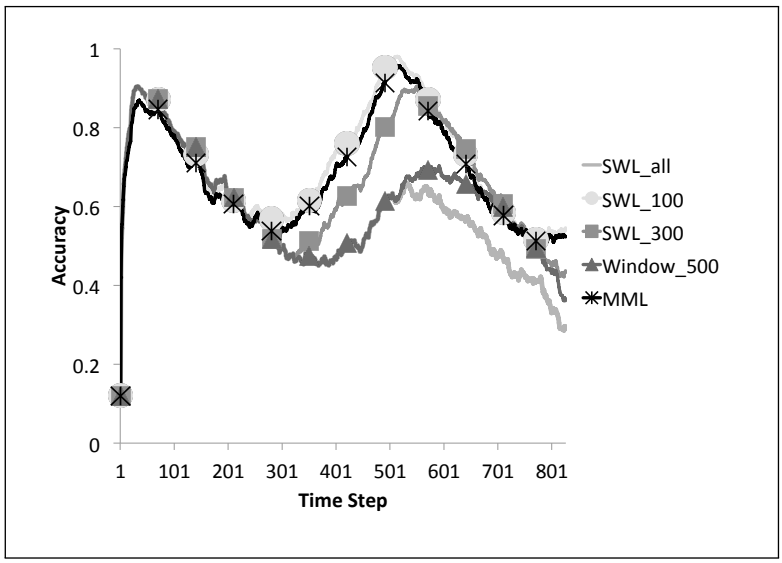

FIGURE 12. Gradual Changes

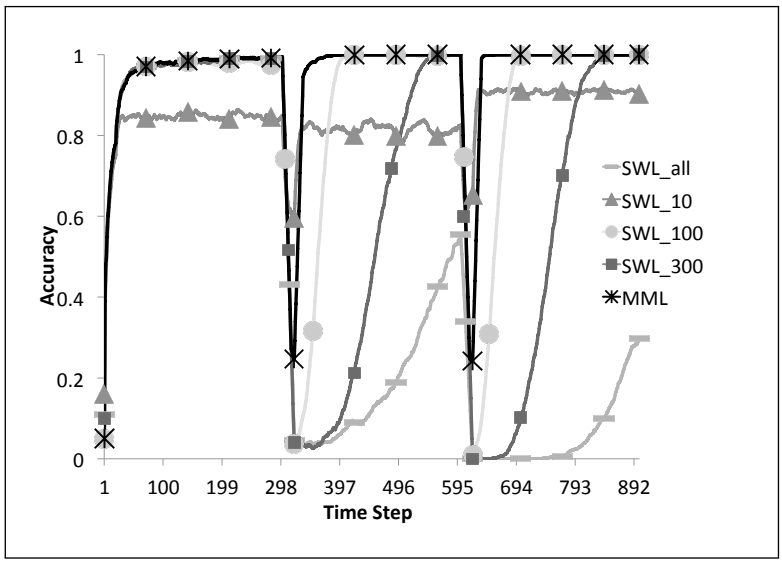

FIGURE 13. Abrupt Changes - Encountered New Behaviour

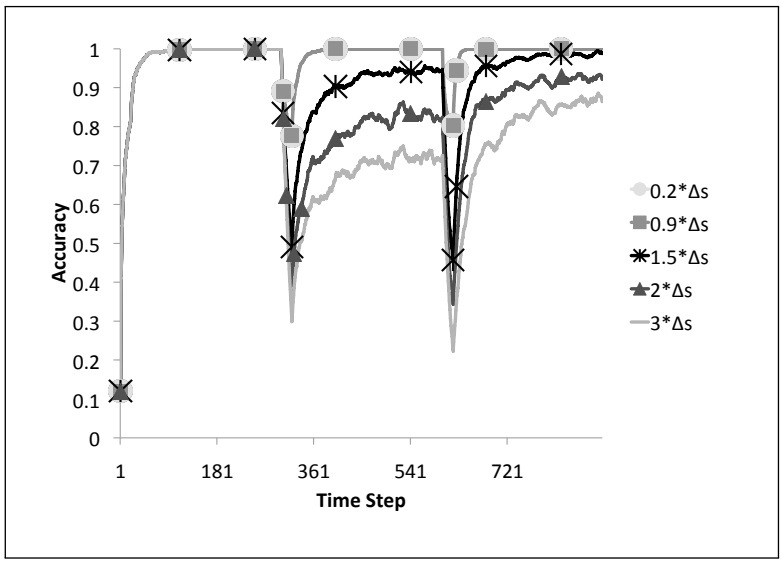

FIGURE 14. Abrupt Changes - Effect of CDT

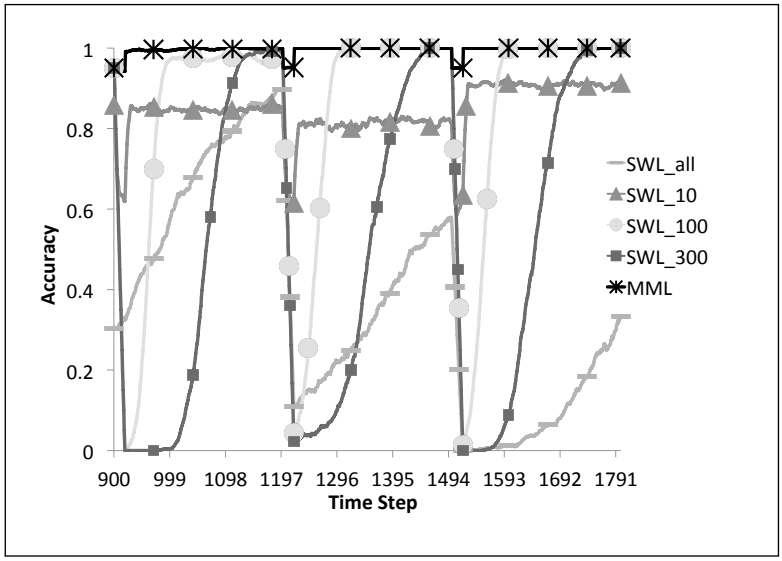

FIGURE 15. Abrupt Changes - Encountered Recurring Behaviour

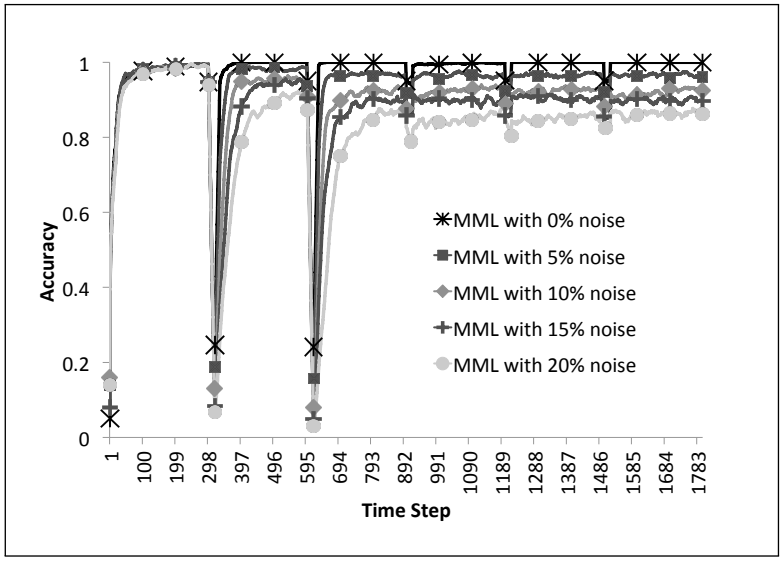

FIGURE 16. Effect of Imperfect Service-side Context

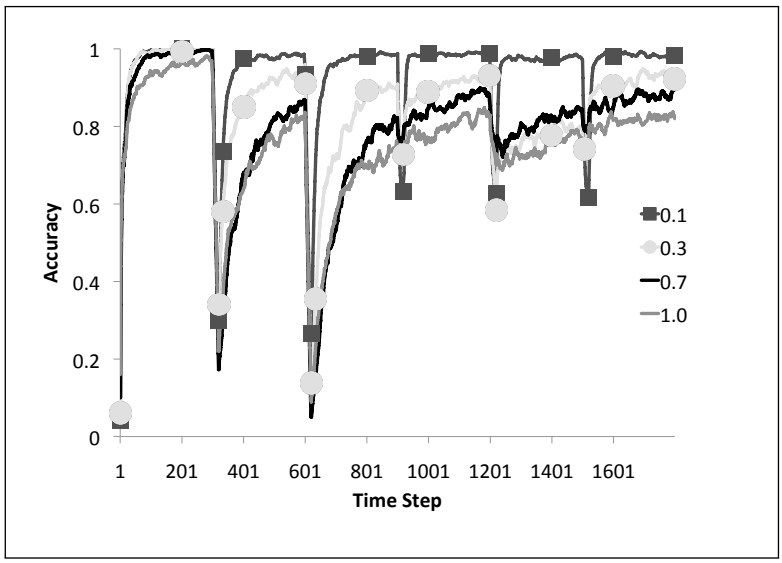

FIGURE 17. Effect of Classifier Weight Learning Factor

TABLE 1. Past user ratings on service s_1 regarding meal texture

User	**Rating**	**Time Step**	**User's Context**
User 1	Hard	t_1	dysphagic
User 2	Soft	t_2	no chewing or swallowing difficulties
User 3	Soft	t_3	no chewing or swallowing difficulties

TABLE 2. Past user ratings on service s_1 regarding delivery time

User	**Rating**	**Time Step**	**User's Context**
User 1	30 minutes	t_1	in an urban location
User 2	2 hours	t_2	in a rural location
User 3	30 minutes	t_3	in an urban location

TABLE 3. Past user ratings on service s_2 regarding meal taste

Rating	**Time Step**	**Service's Context**
5 stars	t_1	Recipe 1
5 stars	t_2	Recipe 1
	...	
5 stars	t_{20}	Recipe 1
1 star	t_{21}	Recipe 2
1 star	t_{22}	Recipe 2
1 star	t_{23}	Recipe 2

TABLE 4. Notation used in the paper

Symbol	Description	Symbol	Description
Q	Set of quality attributes	M	A model's body
C^u	Set of user-side context attributes	$qval$	Quality value prediction function
C^s	Set of service-side context attributes	c_s	Service-related context attribute
ctx_u	A quality attribute's user-side context attributes	c_u	User-related context attribute
ctx_s	A quality attribute's service-side context attributes	$dist$	Distance between two context values
dom	An attribute's domain	sim	Similarity between two context conditions
v_q	A quality value	δ_{wt}	Learning rate of model weights
$\vec{v_s}$	Vector of values for service-side context	$thrsh$	Model weight threshold
$\vec{v_u}$	Vector of values for user-side context	NCT	New classifier threshold
Ω	Model library	CDT	Context dissimilarity threshold
wt	A model's weight	Δ_s	Abrupt mean shift for a context attribute
ψ	A model's precondition		

TABLE 5. Experimental Settings

Description	**Value**
Number of runs	100
Model weight's learning rate, δ_{wt}	0.7
Model similarity threshold , $thrsh$	0.5
Number of observations for a stable model, $stabilitySize$	15
Number of user-side context attributes	3
Number of service-side context attributes	1
Number of possible quality outcomes	5
Number of generating sources for abrupt and gradual changes	3
Abrupt mean shift for a context attribute, Δ_s	10